



INDICE

1.- Instalación de los paquetes de la API de Comunicaciones de Java en la JVM Instalada en Windows.	1
2.- Instalación de los paquetes de la API de Comunicaciones de Java en la JVM Instalada en Linux.	2
3.- Librería de clase TfhkaJava.jar	4
4.- Métodos básicos de la Clase Tfhka	5
4.1- <i>BOOLEAN OpenFpctrl (String lpPortName)</i>	5
4.2- <i>VOID CloseFpctrl ()</i>	6
4.3- <i>BOOLEAN CheckFprinter ()</i>	6
4.4- <i>BOOLEAN ReadFpStatus ()</i>	6
4.5- <i>BOOLEAN SendCmd (String cmd)</i>	7
4.6- <i>INT SendCmdFile (String file)</i>	7
4.7- <i>BOOLEAN UploadReportCmd (String cmd, String file)</i>	7
4.8- <i>BOOLEAN UploadStatusCmd (String cmd, String file)</i>	8
5.- Atributos y métodos públicos orientados a objetos de la clase Tfhka.	8
5.1- <i>Atributos de objetos.</i>	8
5.2- <i>Métodos de objetos.</i>	11
6.- Anexos	14
Anexo 1: Información del Status de la Impresora Fiscal	14
Anexo 2: Comandos Para Leer el Estado de la Impresora	16
Anexo 3: Aplicación Demo en Java para Impresoras Fiscales "FacturadorJava" bajo Windows	21
Anexo 4: Aplicación Demo en Java para Impresoras Fiscales "DemoTfhkaJava" bajo Windows	22
Anexo 5: Aplicación Demo en Java para Impresoras Fiscales "DemoJavaTfhka" bajo Linux	23

1.- INSTALACION DE LOS PAQUETES DE LA API DE COMUNICACIONES DE JAVA EN LA JVM INSTALADA EN WINDOWS.

La plataforma de desarrollo java tiene un conjunto de archivos requeridos para establecer comunicación con el puerto serial de la PC. Estos archivos están agrupados en una carpeta comprimida que se denomina “RXTXCommApi”, en donde se encuentran clasificados los archivos necesarios para que una aplicación java pueda escribir y leer en el puerto serial.

El RXTXCommApi de java es una librería propia de la JRE y de la JDK que esta disponible para descargarla desde Sun Microsystems, las cuales tienen definidas para un sistema operativo determinado, es decir; el RXTXCommApi de Windows es solo para aplicaciones desarrolladas en ambiente java bajo Windows, ya que la JVM se instala en la PC de diversas formas dependiendo del sistema operativo y el RXTXCommApi de Windows trae un archivo .dll. Si se trabaja en otro sistema operativo como MAC, LINUX, SOLARIS; se debe descargar el RXTXCommApi respectivo para cada uno de ellos.

En el RXTXCommApi para Windows hay dos carpetas que contienen archivos. El nombre de la carpeta indica en donde se deben pegar los mismos hacia la jre ó jdk/jre de la JVM instalada en este sistema operativo.

Lo primero que se debe hacer antes de copiar los archivos, es verificar si el Windows tiene el Java instalado en C:/Archivo de Programas. De no tenerla se debe instalar ó descargar la versión más reciente de JDK ó JRE en su PC.

Una vez determinada la JVM de su PC debe hacer lo siguiente:

- 1- Copiar los archivos rxtxSerial.dll y rxtxParallel.dll en la siguiente ruta de su JVM: “C:/Archivo de Programas/Java/jdk/jre/bin”. Si su JVM es solo jre, se copia el archivo en: “C:/Archivo de Programas/Java/jre/bin”.
- 2- Copiar el archivo RXTXcomm.jar en la siguiente ruta de su PC: “C:/Archivo de Programas/Java/jdk/jre/lib/ext”. Si su JVM es solo jre, se copia el archivo en: “C:/Archivo de Programas/Java/jre/lib/ext”.

Ahora su equipo tiene la JVM configurada para escribir y leer en el puerto serial.

2.- INSTALACION DE LOS PAQUETES DE LA API DE COMUNICACIONES DE JAVA EN LA JVM INSTALADA EN LINUX.

En el RXTXCommApi para Linux hay tres carpetas que contienen los archivos de manera distribuida. Estos archivos son: RXTXcomm.jar, librxtxParallel.so y librxtxSerial.so.

Lo primero que se debe hacer antes de copiar los archivos, es verificar si su distribución de Linux tiene el Java instalado en la ruta: /usr/lib/. De no tenerla se debe instalar ó descargar la versión más reciente de JDK ó JRE para su distribución.

Una vez determinada la JVM de su PC debe hacer lo siguiente:

- 1- Copiar los archivos librxtxParallel.so y librxtxSerial.so en la siguiente ruta de su JVM:

```
% setenv LD_LIBRARY_PATH `pwd`: $LD_LIBRARY_PATH
$ export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH
```

Si tienes privilegios en tu máquina o inicias como super usuario root:

```
Copiar librxtxSerial.so y librxtxParallel.so en: /usr/lib/ <JDK>/jre/lib/i386
Copiar librxtxSerial.so y librxtxParallel.so en: /usr/lib/ <JDK>/jre/lib/i386/client
```

- 2- Copiar el archivo RXTXcomm.jar en la siguiente ruta de su PC:

Si está utilizando JDK (no JRE) RXTXcomm.jar Añadir a su CLASSPATH.

Ejemplo: Si usted no tiene un CLASSPATH establecido actualmente,

```
Setenv% CLASSPATH »pwd` / RXTXcomm.jar
```

ó, si usted tiene algo en su CLASSPATH ya,

```
Setenv% CLASSPATH »pwd` /RXTXcomm.jar: $ CLASSPATH
```

Si tienes privilegios en tu máquina o inicias como super usuario root:

```
Copiar RXTXcomm.jar en /usr/lib/jvm/<JDK>/jre/lib/ext
```

Lo último que se debe realizar es la configuración del juego de caracteres de Linux. Por defecto la codificación del juego de caracteres en LINUX viene en UTF-8, lo que genera inconveniente para el envío de ASCII extendidos al puerto serial, ya que los rechaza y no los procesa estando armada la trama de caracteres correctamente. Para ello se debe cambiar la configuración de los caracteres a ISO-8859-15 mediante la edición de estos archivos de la siguiente manera:

1. Cambiar el RC_LANG del archivo /etc/sysconfig/language.

```
vi /etc/sysconfig/language
```

```
RC_LANG="de_DE@euro"
```

2. Cambiar el CONSOLE_ENCODING del archivo /etc/sysconfig/console

```
vi /etc/sysconfig/console
```

```
CONSOLE_ENCODING="ISO-8859-15"
```

3. Agregar estas tres líneas de códigos en el archivo /etc/bash.bashrc

```
vi /etc/bash.bashrc.local
```

```
export LANG=de_DE@euro  
export LC_MESSAGES=POSIX  
export LC_CTYPE=de_DE@euro
```

4. Reinicie el sistema para que los cambios de la codificación de los caracteres tengan efecto.

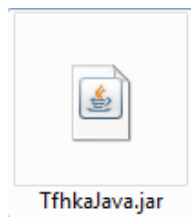
NOTA 1: Se debe tratar de no poner en los ficheros fuente (*.java) caracteres extendidos, ya que al cambiar la configuración los cambia por otros caracteres. Ejm. Nombre y valores asignados de variables que contengan caracteres como ñ, á, é, Ñ, ¡, í, ó, ú, Á, É, Í, Ó, Ú, etc..

NOTA 2: También puede cambiar la codificación de caracteres en las propiedades del entorno de desarrollo Java que emplee. Ejm, En Eclipse es en menú Windows-> Preferences -> General -> Content Types -> Text -> JavaSource

Ahora su equipo tiene la JVM configurada para escribir y leer en el puerto serial.

3.- LIBRERÍA DE CLASE TfhkaJava.jar

La librería de clase TfhkaJava.jar es un archivo comprimido para implementarlo en un ambiente de desarrollo java, el cual contiene una clase llamada tfhka.Tfhka.class que permite manipular el puerto serial para el envío y recepción de tramas ASCII, con el fin de hacer la integración de las Impresoras Fiscales desarrolladas en la compañía The Factory HKA C.A. con aplicaciones desarrolladas en java. Este archivo es multiplataforma, se puede emplear en Linux, Solaris, Windows y otros sistemas operativos que soporte java, siempre y cuando tengan implementado el CommApi correspondiente a su tipo en la JVM.



El objetivo del archivo .jar Tfhkajava es reducir el protocolo de comunicación de tramas ASCII a la impresora fiscal en proyectos desarrollado en java, ya que hasta ahora se los programas facturadores desarrollado en diversas plataforma emplean el protocolo directo de comunicación.

Importación y declaración de la clase Tfhka

Para hacer uso de esta librería en su proyecto, ya sea de entorno Eclipse, NetBeans ú otro desarrollador Java debe importarla de su ubicación. De igual manera se debe importar el archivo RXTXcomm.jar de su JVM (...jre/lib ó ...jdk/jre/lib) en caso que no aparezca la referencia por defecto en librares.

Una vez agregada las librerías TfhkaJavaPa.jar y comm.jar a su Proyecto Java, se debe hace la importación respectiva de la clase del archivo jar importado "Tfhka" en la(s) clase(s) que lo requieran de su proyecto. Esto se hace de esta manera en el encabezado de la clase solicitante:

```
import tfhka.Tfhka;
```

Hecho esto se puede empezar ha trabajar con la clase de Tfhka, con sus atributos y métodos públicos.



1-) Declaración de un objeto tipo Tfhka.

<Tipo de modificador> Tfhka <Nombre del Objeto>

Ejemplos:

```
private    Tfhka  Mi_Enlace;
```

```
public    Tfhka  Mi_Ruta;
```

2-) Inicialización del objeto Tfhka.

La clase Tfhka tiene un solo constructor que recibe como parámetro un String ó cadena de carácter con el nombre del puerto serial ó terminar serial.

<Nombre del Objeto> = new Tfhka(<Nombre del Puerto serial>)

Ejemplos:

En Windows:

```
Mi_Enlace = new Tfhka ( "COM1" );
```

En Linux:

```
Mi_Ruta = new Tfhka ( "/dev/ttyS0" );
```

Si el parámetro nombre de puerto está correctamente escrito, al efectuar esta inicialización se define el *Terminar* de la clase ó String nombre de puerto, se abre si no está en uso y se define el estado del mismo por medio de la variable global booleana *IndPuerto*, donde se pone en *true* si el puerto esta abierto y en *false* si el puerto está cerrado o no se pudo abrir.

4.- MÉTODOS DE LA CLASE Tfhka

4.1- BOOLEAN *OpenFpctrl* (String *lpPortName*)

Parámetro: Puerto COM (Ej. "COM1" o "COM2") ó Terminar "../term/" ("/dev/term/b")

Método: Apertura del Puerto Serial "COM" ó Terminar "../term/".

Retorno: Puerto Abierto = *true* ó Falla de Apertura = *false*.

Este método se ejecuta inicialmente en el constructor único de la clase. Se puede volver a ejecutar si se requiere.

Ejemplo:

// Cambio de puerto serial a COM4

```
boolean port_status = Mi_Enlace.OpenFpctrl ("COM4");
```

4.2- VOID CloseFpctrl ()

Es el método de la clase que cierra el puerto serial establecido de la misma. Pone a la variable booleana *IndPuerto* = *false*.

Se ejecuta de esta manera: *Mi_Enlace.CloseFpctrl ();*

4.3- BOOLEAN CheckFprinter ()

Método: Verifica si la impresora está Conectada.

Retorno: Impresora Conectada = *true* ó Impresora Desconectada = *false*.

Se ejecuta de esta manera: *Mi_Enlace.CheckFprinter ();*

4.4.- BOOLEAN ReadFpStatus ()

Método: Lectura Referente a la Información del Status & Error de la Impresora Fiscal

Retorno: Stand-by = *true* ó Error = *false*.

Se ejecuta de esta manera: *Mi_Enlace.ReadFpStatus ();*

Al llamar a este método se establece el valor de la variable String *Estado*. (Ver Anexo 1)

Ejemplo:

El estado de la impresora en modo de espera y sin error sería así:

```
Mi_Enlace.Estado = "Status: 04 Error: 00";
```

4.5- BOOLEAN SendCmd (String cmd)

Parámetro: Comando cmd: Trama de caracteres ASCII. Ejm: "p-1200"

Método: Envía una Línea de Comando a la Impresora Fiscal.

Retorno: Comando Enviado = **true** ó Comando Invalido = **false**.

Ejemplo: Para hacer un reporte Z, se ejecuta de esta manera:

```
Mi_Enlace.SendCmd ("10Z");
```

Al ejecutar este método se establece el estado del envío por medio del String *Estado*.

```
Mi_Enlace.Estado = "Status: 00 Error: 00";
```

4.6- INT SendCmdFile (String file)

Parámetro: Ruta ó Archivo: Ubicación del Archivo.txt ó Archivo.dat.

Método: Envía Comandos a la Impresora Fiscal leído de un archivo de texto plano.

Retorno: Número de comandos válidos.

Ejemplo:

```
Mi_Enlace.SendCmdFile ("C:\MisFiles\Factura1.txt");
```

Al ejecutar este método se realiza envío por medio de los comandos leídos línea por línea del archivo "Factura1.txt".

4.7- BOOLEAN UploadReportCmd (String cmd, String file)

Parámetros:

Comando cmd: Comando de Reporte a cargar. Ejm: "U0X"

Ruta ó Archivo: Archivo.txt ó .dat en donde se copiara la información del reporte.

Método: Envía los datos de un Reporte al PC y lo escribe en un archivo de texto plano.

Retorno: Stanby = **true** Error = **false**.

Ejemplo:

```
Mi_Enlace.UploadReportCmd ("U0Z", "C:\MisFiles\ReporteZ.txt");
```

Al ejecutar este método se copia en el archivo "ReporteZ.txt" la información cargada de un Reporte Z. (Ver Anexo 2)

4.8- *BOOLEAN UploadStatusCmd (String cmd, String file)*

Parámetros:

Comando cmd: Estado a subir. Ejm: "S1"

Ruta ó Archivo: Archivo.txt ó .dat en donde se copiara la trama que se subirá.

Método: Envía una trama de información al PC del estado leído de la impresora y lo escribe en un archivo de texto plano.

Retorno: Stanby = **true** Error = **false**.

Ejemplo:

```
Mi_Enlace.UploadReportCmd ( "S2" , "C:\MisFiles\Status1.txt");
```

Al ejecutar este método se lee de la impresora fiscal el estado "S2" y se sube al PC para copiar la trama de información respectiva en el archivo "Status1.txt". (Ver Anexo 2)

5-. Atributos y métodos públicos orientados a objetos de la clase Tfhka.

5.1- Atributos de objetos:

- *StatusErrorPrinter* tipo **PrinterStatus**: Representa una estructura que establece y retorna los parámetros de Status y Error de la impresora fiscal cuando se ejecuta el método correspondiente. Sus elementos son los siguientes:

int printerStatusCode: Valor entero del Status.

int printerErrorCode: Valor entero del Error.

String printerStatusdescripcion: Descripción del Status.

String printerErrorDescripcion: Descripción del Error.

boolean ErrorValidity: Indica si Error de la impresora es valido.

- *ReportePC* tipo **ReportData**: Representa una estructura que contiene todos los datos de los reportes X y Z cuando se envían al PC con los métodos correspondientes. Sus elementos son los siguientes:

int getNumberOfLastZReport(): Número del ultimo reporte Z.

Date getZReportDate(): Fecha del último reporte Z.

int getNumberOfLastInvoice(): Número de la última factura.

Date getLastInvoiceDate(): Fecha y hora de la última factura.

double getFreeSalesTax(): Monto Exento.

double getTotalVenta(): Monto Total Neto de la Venta.

double generalRate1Tax: Monto del IVA General ó tasa 1.

double reducedRate2Tax: Monto del IVA Reducido ó tasa 2.

double additionalRate3Tax: Monto del IVA Adicional ó tasa 3.

double getTotalVentaDevolution(): Monto Total Neto en Devolución.
double getTotalVentaDebitNote(): Monto Total Neto en Nota Débito.
int getNumberOfLastCreditNote(): Número de última Nota de Crédito.
int getOfLastDebitNote(): Número de última Nota de Débito.
int getNumberOfLastDNF(): Número del último Documento No Fiscal.

- *S1Estado* tipo **S1PrinterData**: Representa la estructura de datos del estado S1 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

int getCashierNumber(): Número de cajero asignado.
double getTotalDailySales(): Monto total de ventas diaria.
int getLastInvoiceNumber(): Número de la última factura emitida.
int getQuantityOfInvoicesToday(): Cantidad de facturas en el día.
int getLastNCNumber(): Número de la última Nota de Crédito.
int getQuantityOfNCsToday(): Cantidad de Notas Créditos en el día.
int getLastNDNumber(): Número de la última Nota de Débito.
int getQuantityOfNDsToday(): Cantidad de Notas Débitos en el día.
int getNumberNonFiscalDocuments(): Número del último DNF.
int getQuantityNonFiscalDocuments(): Cantidad de DNF.
int getDailyClosureCounter(): Contador de Cierre Diario.
int getAuditReportsCounter(): Contador de reporte de auditoría.
String getRUC(): RUC de fiscalización de la impresora fiscal.
String getDV(): DV de Fiscalización.
String getRegisteredMachineNumber(): Serial de la impresora fiscal.
Date getCurrentPrinterDateTime(): Fecha y hora de la impresora.

- *S2Estado* tipo **S2PrinterData**: Representa la estructura de datos del estado S2 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

double subTotalBases: Monto sub-total de las Bases Imponibles.
double sudTotalTax: Monto sub-total de IVA.
String dataDummy: Data de relleno.
double amountPayable: Monto por pagar.
int numberPaymentsMade: Cantiidad de pagos realizados.
int condition: Condición de transacción.

- *S3Estado* tipo **S3PrinterData**: Representa la estructura de datos del estado S3 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

int typeTax1: Tipo de tasa 1 (Modo Incluido = 1, Modo Excluido = 2).
double tax1: Valor de la tasa 1 en porcentaje (%).

int typeTax2: Tipo de tasa 2 (Modo Incluido = 1, Modo Excluido = 2).
double tax2: Valor de la tasa 2 en porcentaje (%).
int typeTax3: Tipo de tasa 3 (Modo Incluido = 1, Modo Excluido = 2).
double tax3: Valor de la tasa 3 en porcentaje (%).
int[] systemFlags0to19: Lista de valores de los flags del 1 al 19.

- *S4Estado* tipo **S4PrinterData**: Representa la estructura de datos del estado S4 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

double[] listCumulaMountsMeansPayments: Lista de los montos acumulados de los 16 medios de pagos de la impresora fiscal.

- *S5Estado* tipo **S5PrinterData**: Representa la estructura de datos del estado S5 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

String RUC: RUC de fiscalización de la impresora.
String DV: DV de fiscalización de la impresora.
String registeredMachineNumber: Serial de la impresora fiscal.
int numberMemoryAudit: Número de memoria de auditoría.
double capacityTotalMemoryAudit: Capacidad de la memoria de auditoría en MB.
double disponyCapacityMemoryAudit: Disponibilidad de la memoria de auditoría en MB.
int numberDocumentRegisters: Cantidad de documentos registrados en la memoria de auditoría.

- *S6Estado* tipo **S6PrinterData**: Representa la estructura de datos del estado S6 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

String getBit_Facturacion(): Indica la presencia de papel en la estación de facturación.
String getBit_Slip(): Indica la presencia de papel en la estación de Slip/Chequera.
String getBit_Validacion(): Indica la presencia de papel en la estación de Validación.

- *S7Estado* tipo **S7PrinterData**: Representa la estructura de datos del estado S7 que se establece cuando se sube al PC con su método correspondiente. Sus elementos son los siguientes:

String getMICR(): Lectura del MICR del cheque.

- *ReporteArrayPC* tipo **ReportData[]**: Representa una lista de objetos tipo *ReportData* que se carga cuando se hace lectura de memoria fiscal con los métodos correspondientes, con una dimensión de un máximo de 2000 elementos posibles.

- **PrinterException**: Representa un tipo de excepción que arrojan algunos métodos de creación de objetos de las estructuras anteriormente definidas cuando ocurre un error en la transacción con la impresora fiscal. Está compuesto por los siguientes elementos:

PrinterStatus *StatusError*: Objeto que contiene información del Status y el Error al momento de generarse la excepción.

getMessage(): Contiene la descripción de la excepción arrojada.

5.2- Métodos de Objetos:

5.2.1- *PrinterStatus* ***getPrinterStatus()***.

Método: Establece los datos del objeto *StatusErrorPrinter* de la clase por medio de un llamado interno al método *ReadFpStatus ()*.

Retorno: Un objeto de tipo *PrinterStatus* en la variable global *StatusErrorPrinter* de la clase *Tfhka.java* con los valores correspondiente a la repuesta generada.

5.2.2- *ReportData* ***getXReport()*** throws *PrinterExeption*.

Método: Sube al PC un reporte X por medio del comando "U0X" estableciendo sus valores de data en el objeto *ReportePC* de la clase *Tfhka.java*.

Retorno: Un objeto de tipo *ReportData* en la variable global *ReportePC* con toda la información para la carga de un reporte X actual.

Excepción: Arroja la excepción *PrinterExeption*.

5.2.3- *ReportData* ***getZReport()*** throws *PrinterExeption*.

Método: Sube al PC un reporte Z por medio del comando "U0Z" estableciendo sus valores de data en el objeto *ReportePC* de la clase *Tfhka.java*.

Retorno: Un objeto de tipo *ReportData* en la variable global *ReportePC* con toda la información del último reporte Z emitido.

Excepción: Arroja la excepción *PrinterExeption*.



5.2.4- ReportData[] **getZReport**(int StarReportNumber, int EndReportNumber)
throws PrinterException

Parámetros:

- StarReportNumber: Número de reporte Z inicial.
- EndReportNumber: Número de reporte Z final.

Método: Realiza una lectura de memoria fiscal por rango de números y asigna los valores de Z(s) en el objeto *ReporteArrayPC*.

Retorno: Una lista de objetos ReportData en la variable *ReporteArrayPC* con la información de los reportes Z comprendidos en el intervalo solicitado.

Excepción: Arroja la excepción PrinterException.

5.2.5- ReportData[] **getZReport** (Date StarDate, Date EndDate) throws
PrinterException

Parámetros:

- StarDate: Fecha inicial.
- EndDate: Fecha final.

Método: Realiza una lectura de memoria fiscal por rango de fechas y asigna los valores de Z(s) en el objeto *ReporteArrayPC*.

Retorno: Una lista de objetos ReportData en la variable *ReporteArrayPC* con la información de los reportes Z comprendidos en el intervalo solicitado.

Excepción: Arroja la excepción PrinterException.

5.2.6- **printZReport**(Date StarDate, Date EndDate) throws PrinterException

Parámetros:

- StarReportNumber: Número de reporte Z inicial.
- EndReportNumber: Número de reporte Z final.

Método: Imprime una lectura de memoria fiscal por rango de números.

Excepción: Arroja la excepción PrinterException.

5.2.7- **printZReport** (int StarReportNumber, int EndReportNumber) throws
PrinterException.

Parámetros:

- StarDate: Fecha inicial.
- EndDate: Fecha final.

Método: Imprime una lectura de memoria fiscal por rango de fechas.

Excepción: Arroja la excepción PrinterException.



5.2.8- **printXReport()** throws PrinterException.

Método: Imprime un reporte X.

Excepción: Arroja la excepción PrinterException.

5.2.9- **printZReport()** throws PrinterException.

Método: Imprime un reporte Z.

Excepción: Arroja la excepción PrinterException.

5.2.10- S1PrinterData **getS1PrinterData()** throws PrinterException

Método: Sube el estado S1 de la Impresora fiscal al PC y estableces los valores respectivos de la data en el objeto *S1Estado* de la clase Tfhka.java.

Retorno: Un objeto de tipo S1PrinterData en la variable global *S1Estado*.

Excepción: Arroja la excepción PrinterException.

5.2.11- S2PrinterData **getS2PrinterData()** throws PrinterException

Método: Sube el estado S2 de la Impresora fiscal al PC y estableces los valores respectivos de la data en el objeto *S2Estado* de la clase Tfhka.java.

Retorno: Un objeto de tipo S2PrinterData en la variable global *S2Estado*.

Excepción: Arroja la excepción PrinterException.

5.2.12- S3PrinterData **getS3PrinterData()** throws PrinterException

Método: Sube el estado S3 de la Impresora fiscal al PC y estableces los valores respectivos de la data en el objeto *S3Estado* de la clase Tfhka.java.

Retorno: Un objeto de tipo S3PrinterData en la variable global *S3Estado*.

Excepción: Arroja la excepción PrinterException.

5.2.13- S4PrinterData **getS4PrinterData()** throws PrinterException

Método: Sube el estado S4 de la Impresora fiscal al PC y estableces los valores respectivos de la data en el objeto *S4Estado* de la clase Tfhka.java.

Retorno: Un objeto de tipo S4PrinterData en la variable global *S4Estado*.

Excepción: Arroja la excepción PrinterException.

5.2.14- S5PrinterData **getS5PrinterData()** throws PrinterException

Método: Sube el estado S5 de la Impresora fiscal al PC y estableces los valores respectivos de la data en el objeto *S5Estado* de la clase Tfhka.java.

Retorno: Un objeto de tipo S5PrinterData en la variable global *S5Estado*.

Excepción: Arroja la excepción PrinterException.

6.- ANEXOS

Anexo 1

“Información del Status de la Impresora Fiscal”

STATUS		
Retorno (Hex)	Retorno (Decimal)	Comentario
0	0	Status Desconocido
1	1	En Modo Prueba y en Espera
2	2	En Modo Prueba y Emisión de Documentos Fiscales
3	3	En Modo Prueba y Emisión de Documentos No Fiscales
4	4	En Modo Fiscal y en Espera
5	5	En Modo Fiscal y Emisión de Documentos Fiscales
6	6	En Modo Fiscal y Emisión de Documentos No Fiscales
7	7	En Modo Fiscal y Cercana Carga Completa De La Memoria Fiscal Y en Espera
8	8	En Modo Fiscal y Cercana Carga Completa De La Memoria Fiscal Y en Emisión de Documentos Fiscales
9	9	En Modo Fiscal y Cercana Carga Completa De La Memoria Fiscal Y en Emisión de Documentos No Fiscales
0A	10	En Modo Fiscal y Carga Completa De La Memoria Fiscal Y en Espera
0B	11	En Modo Fiscal y Carga Completa De La Memoria Fiscal Y en Emisión de Documentos Fiscales
0C	12	En Modo Fiscal y Carga Completa De La Memoria Fiscal Y en Emisión de Documentos No Fiscales

“Información del Error de la Impresora Fiscal”

Error			
Retorno (Hex)	Retorno (Decimal)	Comentarios	Valido / Invalido
00	0	No hay Error	VALIDO
01	1	Fin en la Entrega de papel	VALIDO
02	2	Error de índole Mecánico en la entrega de Papel	VALIDO
03	3	Fin en la Entrega de papel y Error Mecánico	VALIDO
50	80	Comando Invalido / Valor Invalido	INVALIDO
54	84	Tasa Invalida	INVALIDO
58	88	No hay Asignadas Directivas	INVALIDO
5C	92	Comando Invalido	INVALIDO
60	96	Error Fiscal	INVALIDO
64	100	Error de la Memoria Fiscal	INVALIDO
6C	108	Memoria Fiscal llena	INVALIDO
70	112	Buffer Completo (Debe enviar el Comando de Reinicio)	INVALIDO
80	128	Error en la Comunicación	INVALIDO
89	137	No Hay Respuesta	INVALIDO
90	144	Error LRC	INVALIDO
91	145	Error Interno API	INVALIDO
99	153	Error en la Apertura del Archivo	INVALIDO

Anexo 2.

Comandos Para Leer el Estado de la Impresora

Lectura del Estado 1 (S1)

Este comando permite leer desde el host (PC) el estado de la impresora fiscal, referente a parámetros de la impresora como serial, RIF y datos de factura. Es posible ejecutar este comando en cualquier condición.

Detalle de Data de los 88 bytes de Respuesta de la impresora:

Desde	Hasta	Long	Clase	ITEM
1	2	2	ASCII	Comando S1
3	4	2	ASCII	Numero de cajero asignado
5	21	17	ASCII	Total de ventas diarias(14 bytes)
22	29	8	ASCII	Número última factura
30	34	5	ASCII	Cantidad de facturas en el día
35	42	8	ASCII	Número del documento no fiscal
43	47	5	ASCII	Cantidad de documentos no fiscales
48	51	4	ASCII	Contador de cierres diarios
52	55	4	ASCII	Contador de reportes de auditoria
56	66	11	ASCII	RIF
67	76	10	ASCII	Número de registro del equipo
77	82	6	ASCII	Hora actual en la impresora
83	88	6	ASCII	Fecha actual en la impresora

Lectura del Estado 2 (S2)

Este comando permite leer desde el host (PC) el estado de la factura en curso en transacción. Si es ejecutado este comando sin una factura en curso, los valores obtenidos serán cero.

Detalle de Data de los 69 bytes de Respuesta de la impresora:

Desde	Hasta	Long	Clase	ITEM
1	2	2	ASCII	Comando S2
3	3	1	20h	(carácter blanco)
4	16	13	ASCII	Total de ventas diarias (13 bytes)
17	17	1	20h	(carácter espacio)
18	30	13	ASCII	Número última factura
31	31	1	20h	(carácter espacio)
32	50	19	ASCII	Data Dummy
51	51	1	20h	(carácter espacio)
52	64	13	ASCII	Monto por Pagar
65	68	4	ASCII	Número de Pagos Realizados
69	69	1	ASCII	Condición

Lectura del Estado 3 (S3)

Este comando permite leer desde el host (PC) el estado de la impresora fiscal, referentes a las tasas de impuesto y flags de estado. Es posible ejecutar este comando en cualquier condición.

Detalle de Data de los 57 bytes de Respuesta de la impresora:

Desde	Hasta	SIZE	Clase	ITEM
1	2	2	ASCII	Comando
3	3	1	ASCII	Tipo de tasa 1
4	7	4	ASCII	Valor Tasa 1
8	8	1	ASCII	Tipo de tasa 2
9	12	4	ASCII	Valor Tasa 2
13	13	1	ASCII	Tipo de tasa 3
14	17	4	ASCII	Valor Tasa 3
18	57	40	ASCII	System Flags 1-20 (Cada Flag tiene 2 caracteres)

Lectura del Estado 4 (S4)

Este comando permite leer desde el host (PC) el estado de la impresora fiscal, referentes a los Medios de Pago. Es posible ejecutar este comando en cualquier condición.

Detalle de Data de los 162 bytes de Respuesta de la impresora:

Desde	Hasta	Long	Clase	ITEM
1	2	2	ASCII	Comando S4
3	12	10	ASCII	Medio de Pago 1
13	22	10	ASCII	Medio de Pago 2
23	32	10	ASCII	Medio de Pago 3
33	42	10	ASCII	Medio de Pago 4
43	52	10	ASCII	Medio de Pago 5
53	62	10	ASCII	Medio de Pago 6
63	72	10	ASCII	Medio de Pago 7
73	82	10	ASCII	Medio de Pago 8
83	92	10	ASCII	Medio de Pago 9
93	102	10	ASCII	Medio de Pago 10
103	112	10	ASCII	Medio de Pago 11
113	122	10	ASCII	Medio de Pago 12
123	132	10	ASCII	Medio de Pago 13
133	142	10	ASCII	Medio de Pago 14
143	152	10	ASCII	Medio de Pago 15
153	162	10	ASCII	Medio de Pago 16

Comandos Para Leer Reportes de la Impresora

Lectura del Reporte X (U0X)

Este comando permite leer desde el host (PC) el estado de la impresora fiscal, referente al reporte X. Es posible ejecutar este comando en cualquier condición.

Respuesta desde la impresora fiscal:

Desde	Hasta	Long	Clase	ITEM
1	4	4	ASCII	Numero del Ultimo Reporte X
5	10	6	ASCII	Fecha de Reporte X
11	18	8	ASCII	Numero de la Ultima Factura
19	24	6	ASCII	Fecha de la Ultima Factura
25	28	4	ASCII	Hora de la Ultima Factura
29	38	10	ASCII	Ventas Exento
39	48	10	ASCII	Ventas Tasa General (Tasa 1)
49	58	10	ASCII	Impuesto Tasa General (Tasa 1)
59	68	10	ASCII	Ventas Tasa Reducida (Tasa 2)
69	78	10	ASCII	Impuesto Tasa Reducida (Tasa 2)
79	88	10	ASCII	Ventas Tasa Adicional (Tasa 3)
89	98	10	ASCII	Impuesto Tasa Adicional (Tasa 3)
99	108	10	ASCII	Devoluciones Exento
109	118	10	ASCII	Devolución Tasa General
119	128	10	ASCII	Devolución Impuesto Tasa General
129	138	10	ASCII	Devolución Tasa Reducida
139	148	10	ASCII	Devolución Impuesto Tasa Reducida
149	158	10	ASCII	Devolución Tasa Adicional
159	168	10	ASCII	Devolución Impuesto Tasa Adicional
169	176	8	ASCII	Numero de ultima Nota de Crédito

Lectura del Reporte Z (U0Z)

Este comando permite leer desde el host (PC) el estado de la impresora fiscal, referente al reporte Z. Es posible ejecutar este comando en cualquier condición.

Respuesta desde la impresora fiscal:

Desde	Hasta	Long	Clase	ITEM
1	4	4	ASCII	Numero del Ultimo Reporte Z
5	10	6	ASCII	Fecha de Reporte Z
11	18	8	ASCII	Numero de la Ultima Factura
19	24	6	ASCII	Fecha de la Ultima Factura
25	28	4	ASCII	Hora de la Ultima Factura
29	38	10	ASCII	Ventas Exento
39	48	10	ASCII	Ventas Tasa General (Tasa 1)
49	58	10	ASCII	Impuesto Tasa General (Tasa 1)
59	68	10	ASCII	Ventas Tasa Reducida (Tasa 2)
69	78	10	ASCII	Impuesto Tasa Reducida (Tasa 2)
79	88	10	ASCII	Ventas Tasa Adicional (Tasa 3)
89	98	10	ASCII	Impuesto Tasa Adicional (Tasa 3)
99	108	10	ASCII	Devoluciones Exento
109	118	10	ASCII	Devolución Tasa General
119	128	10	ASCII	Devolución Impuesto Tasa General
129	138	10	ASCII	Devolución Tasa Reducida
139	148	10	ASCII	Devolución Impuesto Tasa Reducida
149	158	10	ASCII	Devolución Tasa Adicional
159	168	10	ASCII	Devolución Impuesto Tasa Adicional
169	176	8	ASCII	Numero de ultima Nota de Crédito

Anexo 3.

Aplicación Demo en Java bajo Windows para Impresoras Fiscales “FacturadorJava”

Esta aplicación se ejecuta en Windows haciendo doble clic en el .jar ejecutable, siempre y cuando tengan la JVM instalada y su CommApi correspondiente configurado en la misma.

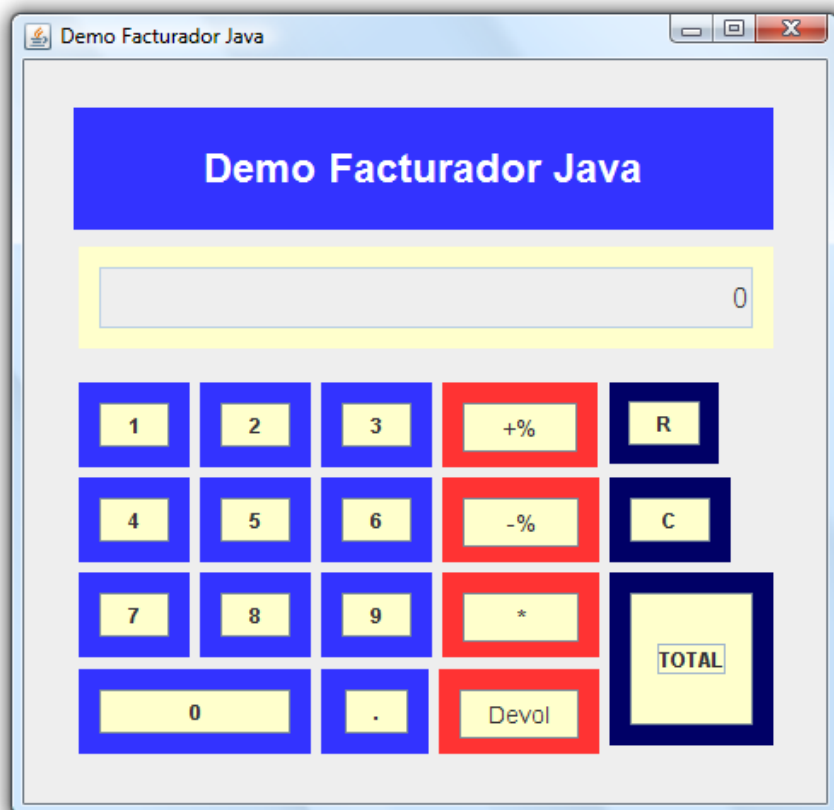
Consiste en un sencillo facturador que maneja la librería TfhkaJavaPa.jar utilizando algunos de sus métodos respectivos, como para abrir puerto serial, enviar comandos para venta a la impresora fiscal y cerrar puerto serial cuando finaliza la aplicación.

Los puertos seriales los asume de manera automática dependiendo si la disponibilidad en el equipo donde se corra la aplicación y también del sistema operativo.

Las tasas de impuestos para el registro de un producto para la venta la toma de la unidad del precio del producto ha registrar, es decir, si se registra un precio de 1,05 Bs. le corresponde la Tasa de Impuesto 1.

Si un producto es unitario se pone su precio y se registra de inmediato, no es necesario multiplicarlo por uno (1).

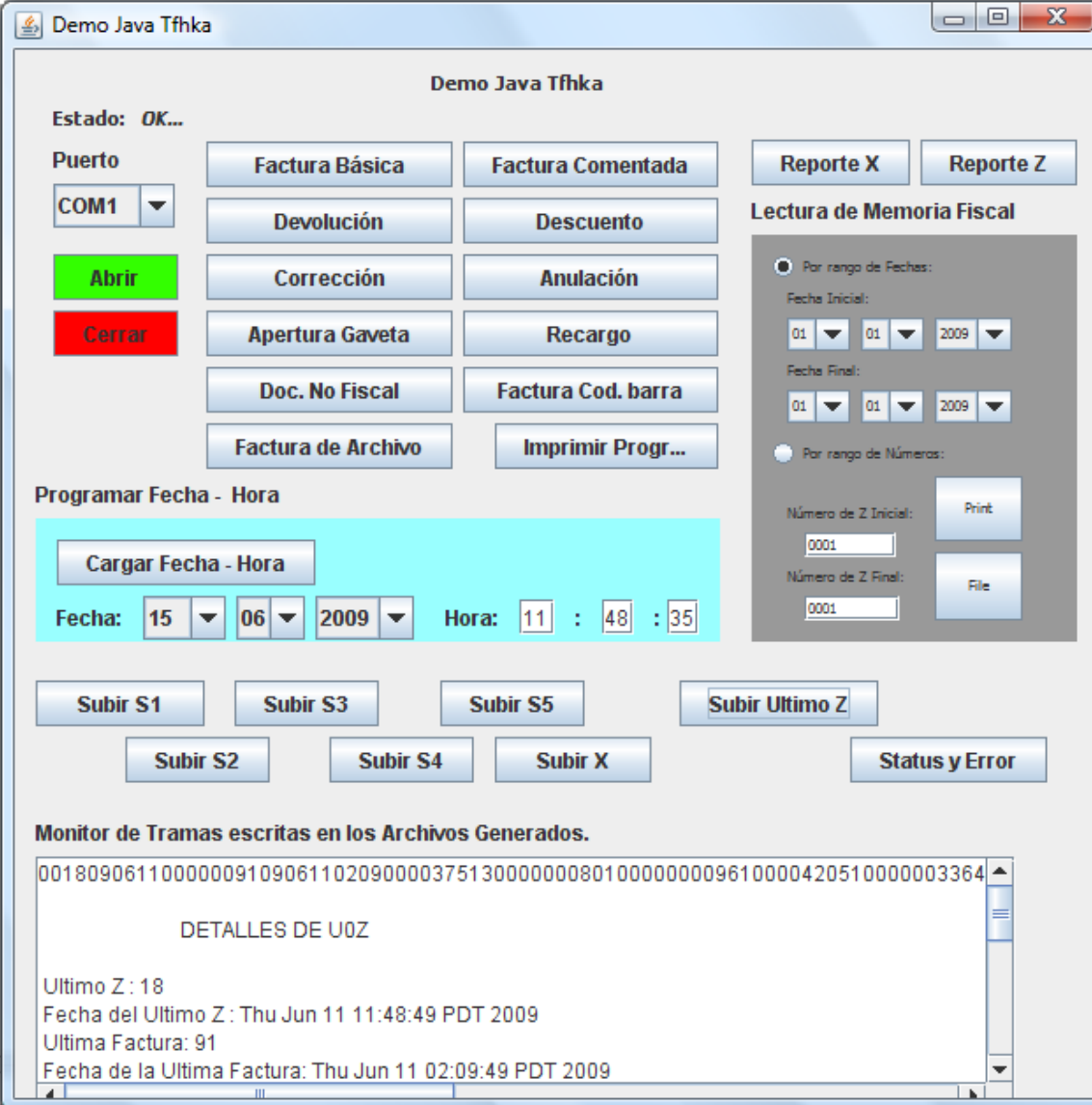
Las operaciones que se pueden hacer en este Demo son: Registro de Producto para la Venta, Registro de Producto para la Devolución, Corrección, Descuentos, Recargos y Totalizar.



Anexo 4.

Aplicación Demo en Java bajo Windows para Impresoras Fiscales “DemoTfhkaJava”

Consiste en una aplicación demo completa que utiliza las funcionalidades más comunes en la impresora fiscal y los métodos de la clase Tfhka.java. Esta desarrollado en NetBeans 5.5 con la JVM JDK 1.6.



The screenshot shows the 'Demo Java Tfhka' application window. It features a menu bar with 'Estado: OK...', 'Puerto' (set to COM1), and 'Abrir' (green) and 'Cerrar' (red) buttons. The main area contains a grid of buttons for fiscal operations: 'Factura Básica', 'Factura Comentada', 'Reporte X', 'Reporte Z', 'Devolución', 'Descuento', 'Corrección', 'Anulación', 'Apertura Gaveta', 'Recargo', 'Doc. No Fiscal', 'Factura Cod. barra', 'Factura de Archivo', and 'Imprimir Progr...'. On the right, there's a 'Lectura de Memoria Fiscal' section with radio buttons for 'Por rango de Fechas' and 'Por rango de Números', each with date and number selection fields and 'Print'/'File' buttons. At the bottom, there's a 'Programar Fecha - Hora' section with a 'Cargar Fecha - Hora' button and date/time pickers. Below that are buttons for 'Subir S1', 'Subir S2', 'Subir S3', 'Subir S4', 'Subir S5', 'Subir Ultimo Z', and 'Status y Error'. The bottom-most section is a 'Monitor de Tramas escritas en los Archivos Generados.' with a text area showing a long hexadecimal string and details of the last Z-closure and invoice.

Anexo 5.

Aplicación Demo en Java bajo Linux para Impresoras Fiscales “FacturadorJava”

Esta aplicación se ejecuta en Linux desde la consola en superusuario mediante la instrucción o línea de comando siguiente:

NombrePCLinux@root#: java -jar /root/Desktop/DemoTFHKAJavaLinux.jar

En este caso el archivo ejecutable de la aplicación está en el escritorio del PC. La aplicación se ejecutará siempre y cuando tengan la JVM instalada y su CommApi correspondiente configurado en la misma.

Consiste en un conjunto de botones que tienen las operaciones básicas que se le envía a la impresora. Este demo también utiliza la librería estándar TfhkaJavaPa.jar.

Al abrir la aplicación se debe elegir el puerto serial y abrirlo, posteriormente se mostrará un mensaje del estatus de la operación de apertura del mismo. Las operaciones de los demás botones se podrán ejecutar solo si la apertura del puerto serial es positiva.

